電視有獎猜題

貧民百萬富翁, 2008

Television quiz shows

ABC 1999: Who wants to be a millionaire

NBC 1964 Merf Griffin: Jeopardy

NBC 1975 Merf Griffin: Wheel of Fortune (word puzzle like hangman)

知識王 app

Slumdog Millionaire

- The player starts with a prize of \$1, and is asked a sequence of n questions.
  - For each question, he may
    - quit and keep his prize
    - answer the question. If wrong, he quits with nothing. If correct, the prize is doubled, and he continues with the next question.
  - After the last question, he quits with his prize.
- The player wants to maximize his expected prize. Once each question is asked, the player is able to assess the probability p that he will be able to answer it. For each question, we assume that p is a random variable uniformly distributed over the range [t, 1], for $0 \le t \le 1$.

Sample Input/Output

- Input
  - Input is a number of lines, each with two numbers: an integer 1 <= n <= 30, and a real 0 <= t <= 1. Input is terminated by a line containing 0 0. This should not be processed.
- Output

  For each input n and t, print the player's expected prize, if he plays the best strategy. Output should be rounded to three fractional digits.

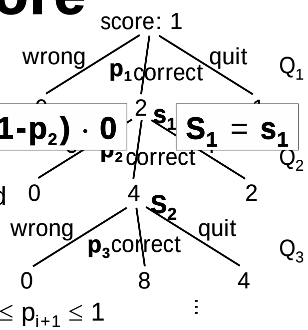
- Sample Input

```
1 0.5
1 0.3
2 0.6
24 0.25
0 0
```

- Sample Output

```
1.500
1.357
2.560
230.138
```


Expected Score



- **Player's best strategy**: **quit** if the score of not answering is larger than the expected score of answering; **answe**

$\mathbf{S_1 = p_2 \cdot S_2 + (1\text{-}p_2) \cdot 0}$ $\mathbf{S_1 = s_1}$

- Let $\mathbf{S_i}$ be the expected score after Q_i is answered
  let $\mathbf{s_i}$ be the current score after Q_i is answered

  expected score $\mathbf{S_i}$ is $\mathbf{s_i}$ if quit answering
  expected score $\mathbf{S_i}$ is $\mathbf{p_{i+1} \cdot S_{i+1} + (1\text{-}p_{i+1}) \cdot 0}$, $t \le p_{i+1} \le 1$

- The best strategy depends on the actual probability $\mathbf{p_{i+1}}$, let $h = s_i / S_{i+1}$

  For $h \ge t$, $\begin{cases} \text{the best strategy is } \textbf{quit} \text{ if } t \le \mathbf{p_{i+1}} < h = s_i / S_{i+1} \ (\mathbf{p_{i+1}} S_{i+1} < s_i) \\ \text{the best strategy is } \textbf{answer} \text{ if } h \le \mathbf{p_{i+1}} \le 1 \ (\mathbf{p_{i+1}} S_{i+1} \ge s_i) \end{cases}$

the expected score after Q_i is answered

For $h < t \le \mathbf{p_{i+1}}$, the best strategy is **answer**

Recursive Version

```
#include <stdio.h>
#include <stdlib.h>
double expectedScore(double s_i, int k, double t) {
    if (k == 0)
        return s_i;
    else {
        double S_ip1 = expectedScore(2*s_i, k-1, t);
        double h = s_i/S_ip1;
        if (t > h)
            h = t;
        return (s_i*(h-t) + S_ip1*(1-h*h)/2) / (1-t);
    }
}
```

```
int main(void) {
    int n;
    double t;
    while (scanf("%d %lf", &n, &t) == 2 && n != 0)
        printf("%.3lf\n", expectedScore(1, n, t));
    system("pause");
    return 0;
}
```


Iterative Version

```
double expectedScore(int n, double t) {
    double s_i, S_ip1, h;
    int k;
    s_i = pow(2, n-1); S_ip1 = s_i*2;
    for (k=n; k>0; k--) { /* k-th question */
        h = s_i / S_ip1;
        if (t > h) h = t;
        S_ip1 = (s_i*(h-t) +
                    S_ip1*(1-h*h)/2) / (1-t);
        s_i /= 2;
    }
    return S_ip1;
}
```